Revista de Educación Matemática
Volumen 32, N°2 (2017), páginas 23 – 34
Unión Matemática Argentina - Famaf (UNC)

COMPRESIÓN DE IMÁGENES: FORMATO JPEG

Javier Lezama

En el número anterior abordamos el método de Hufmann para la compresión de imágenes en el artículo "Compresión de Imágenes: Codificación de Huffman". En el presente trabajo nos ocuparemos de la compresión de imágenes JPEG y como interviene la codificación de Hufmann en dicho proceso.

En 1992, el formato JPEG se convirtió en un estándar internacional para la compresión de imágenes digitales. El acrónimo JPEG proviene de Joint Photographic Experts Group. JPEG se formó en la década de 1980 por los miembros de la Organización Internacional de Normalización y la Unión Internacional de Telecomunicaciones (ISO y UIT en inglés, respectivamente). De acuerdo con un artículo del año 2000 [1], más del 80 % de todas las imágenes que se transmiten a través de Internet se almacenan utilizando el estándar JPEG. A pesar de la popularidad de la norma, los miembros del JPEG rápidamente identificaron algunos problemas con el formato e hicieron una lista de las mejoras que se debían incluir en la próxima generación del formato.

En este artìculo, vamos a describir muy básicamente el algoritmo JPEG, ver cómo funciona por medio de un ejemplo, discutir el proceso de inversión, y cómo interviene la codificación de Hufmann. Finalmente, describiremos los ventajas y problemas que presenta este formato.

El algoritmo JPEG tiene muchas características que no están cubiertas aquí. La intención de este trabajo es proporcionar una idea muy básica de cómo JPEG puede ser utilizado para comprimir una imagen digital. Hay cuatro pasos básicos en el algoritmo - preprocesamiento, transformación, cuantificación y codificación. Ya hemos hablado de la codificación de Huffman, en la parte uno de este trabajo (ver [5]), y JPEG utiliza una versión más sofisticada de la codificación de Huffman para llevar a cabo la misma. No vamos a discutir los métodos de codificación avanzadas utilizadas por JPEG aquí. El lector interesado puede consultar [2].

Para explicar los pasos del algoritmo, utilizaremos como ejemplo de ejecución, la imagen de la Figura 1. Las dimensión de la imagen es de 160×240 pixeles.

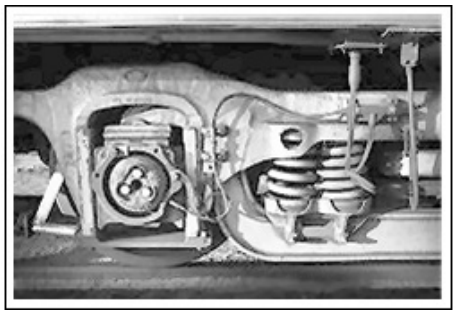


Figura 1. Imagen original.

§1. Preprocesamiento

Dada una imagen en escala de grises lo primero que realizamos es una partición de la misma en bloques de 8×8 pixeles. Si las dimensiones de la imagen no son divisibles por 8, los pixeles sobrantes requieren un tratamiento más particular, pero por motivos de presentación, hemos optado por una imagen de dimensiones son $160 \times 240 = 8 \times 8 \times 20 \times 30$. Por lo que tenemos $20 \times 30 = 600$ bloques (ver Figura 2).

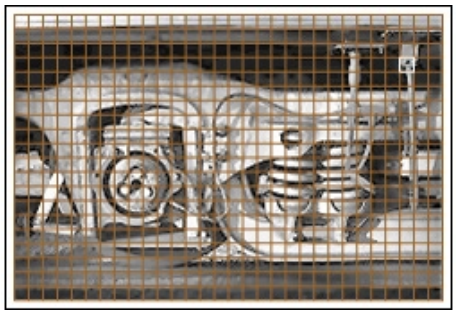


Figura 2. Imagen original en bloques.

Para mostrar la matemática detrás del formato JPEG vamos a trabajar con el bloque de tamaño 8×8 correspondiente al bloque ubicado en la fila 4 y columna 28 de la división en bloques de la imagen anterior (ver Figura 2).

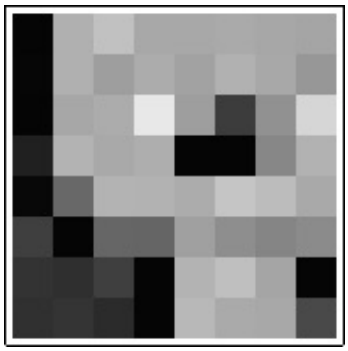


Figura 3. Bloque 8 x 8.

A continuación mostramos la matriz 8×8 que representa la intensidad de grises de la imagen anterior (ver Figura 3) con valores entre 0 (negro) y 255 (blanco)

$$\begin{pmatrix} 5 & 176 & 193 & 168 & 168 & 170 & 167 & 165 \\ 6 & 176 & 158 & 172 & 162 & 177 & 168 & 151 \\ 5 & 167 & 172 & 232 & 158 & 61 & 145 & 214 \\ 33 & 179 & 169 & 174 & 5 & 5 & 135 & 178 \\ 8 & 104 & 180 & 178 & 172 & 197 & 188 & 169 \\ 63 & 5 & 102 & 101 & 160 & 142 & 133 & 139 \\ 51 & 47 & 63 & 5 & 180 & 191 & 165 & 5 \\ 49 & 53 & 43 & 5 & 184 & 170 & 168 & 74 \end{pmatrix}$$

Valores de intensidad de los pixeles del bloque de la Figura 3.

La última parte del procesamiento de la imagen es restar 127 a cada intensidad de los pixeles en cada bloque. Este paso centra las intensidades sobre el valor 0 y tiene la finalidad de simplificar las matemáticas en las etapas de transformación y de cuantificación. Para nuestro ejemplo realizamos la resta y obtenemos los nuevos valores. La siguiente matriz es el bloque resultante de realizar la resta anteriormente explicada al ejemplo en cuestión

$$\begin{pmatrix} -122 & 49 & 66 & 41 & 41 & 43 & 40 & 38 \\ -121 & 49 & 31 & 45 & 35 & 50 & 41 & 24 \\ -122 & 40 & 45 & 105 & 31 & -66 & 18 & 87 \\ -94 & 52 & 42 & 47 & -122 & -122 & 8 & 51 \\ -119 & -23 & 53 & 51 & 45 & 70 & 61 & 42 \\ -64 & -122 & -25 & -26 & 33 & 15 & 6 & 12 \\ -76 & -80 & -64 & -122 & 53 & 64 & 38 & -122 \\ -78 & -74 & -84 & -122 & 57 & 43 & 41 & -53 \end{pmatrix}$$

Bloque resultante después de la resta.

§2. Transformación

El formato de compresión estándar de imagen JPEG utiliza la transformada del coseno discreta (DCT por discrete cosine transformation) dada por la matriz C siguiente

$$C = \frac{1}{2}\begin{pmatrix} \cos(\frac{\pi}{4}) & \cos(\frac{\pi}{4}) & \cos(\frac{\pi}{4}) & \cos(\frac{\pi}{4}) & \cos(\frac{\pi}{4}) & \cos(\frac{\pi}{4}) & \cos(\frac{\pi}{4}) & \cos(\frac{\pi}{4}) \\ \cos(\frac{\pi}{16}) & \cos(\frac{3\pi}{16}) & \cos(\frac{5\pi}{16}) & \cos(\frac{7\pi}{16}) & \cos(\frac{9\pi}{16}) & \cos(\frac{11\pi}{16}) & \cos(\frac{13\pi}{16}) & \cos(\frac{15\pi}{16}) \\ \cos(\frac{2\pi}{16}) & \cos(\frac{6\pi}{16}) & \cos(\frac{10\pi}{16}) & \cos(\frac{14\pi}{16}) & \cos(\frac{18\pi}{16}) & \cos(\frac{22\pi}{16}) & \cos(\frac{26\pi}{16}) & \cos(\frac{30\pi}{16}) \\ \cos(\frac{3\pi}{16}) & \cos(\frac{9\pi}{16}) & \cos(\frac{15\pi}{16}) & \cos(\frac{21\pi}{16}) & \cos(\frac{27\pi}{16}) & \cos(\frac{33\pi}{16}) & \cos(\frac{39\pi}{16}) & \cos(\frac{45\pi}{16}) \\ \cos(\frac{4\pi}{16}) & \cos(\frac{12\pi}{16}) & \cos(\frac{20\pi}{16}) & \cos(\frac{28\pi}{16}) & \cos(\frac{36\pi}{16}) & \cos(\frac{44\pi}{16}) & \cos(\frac{52\pi}{16}) & \cos(\frac{60\pi}{16}) \\ \cos(\frac{5\pi}{16}) & \cos(\frac{15\pi}{16}) & \cos(\frac{25\pi}{16}) & \cos(\frac{35\pi}{16}) & \cos(\frac{45\pi}{16}) & \cos(\frac{55\pi}{16}) & \cos(\frac{65\pi}{16}) & \cos(\frac{75\pi}{16}) \\ \cos(\frac{6\pi}{16}) & \cos(\frac{18\pi}{16}) & \cos(\frac{30\pi}{16}) & co(s\frac{42\pi}{16}) & \cos(\frac{54\pi}{16}) & \cos(\frac{66\pi}{16}) & \cos(\frac{78\pi}{16}) & \cos(\frac{90\pi}{16}) \\ \cos(\frac{7\pi}{16}) & \cos(\frac{21\pi}{16}) & \cos(\frac{35\pi}{16}) & \cos(\frac{49\pi}{16}) & \cos(\frac{63\pi}{16}) & \cos(\frac{77\pi}{16}) & \cos(\frac{91\pi}{16}) & \cos(\frac{105\pi}{16}) \end{pmatrix}$$

La matriz C es inversible, más áun C es ortogonal pues $C^t = C^{-1}$ (ver [7]). Cada bloque B de tamaño 8×8 de la imagen preprocesada es multiplicado por la matriz C a izquierda y por C^t a derecha, obteniendo el bloque

$$U = DTC(B) = CBC^t.$$

Analicemos los cambios que realiza la transformada DCT a cada bloque 8×8 de la imagen. Si enumeramos las filas del 0 al 7 de la matriz C, la fila 0 corresponde a la función constante $\frac{\sqrt{2}}{2}$. Sin embargo, para las demás filas C_i se cumple que la suma de los elementos es nula.

Para ver esto, notar que la fila k-ésima, con $1 \leq k \leq 7$, está formada por el coseno de las fracciones

$$\tfrac{k(2j+1)\pi}{16}, \qquad j = 0, \ldots, 7.$$

Usando la fórmula del coseno de la suma $\cos(x+y) = \cos(x)\cos(y) - \operatorname{sen}(x)\operatorname{sen}(y)$, $x, y \in \mathbb{R}$, y que coseno es una función par, i.e. $\cos(-x) = \cos(x)$, tenemos que

$$\cos\left(\tfrac{k(16-(2j+1)\pi}{16}\right) = \cos(k\pi)\cos\left(\tfrac{-(2j+1)\pi}{16}\right) = -\cos\left(\tfrac{(2j+1)\pi}{16}\right)$$

para los k impares. Esto se puede ver de manera informal gráficamente. Por ejemplo, para la fila 1 tenemos

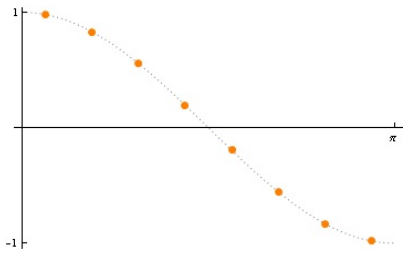


Figura 4. Gráfico de la fila 1.

Para los k pares, de forma análoga, se puede ver que para cualquier i hay un j tal que $\cos(\frac{ki\pi}{16}) = -\cos(\frac{kj\pi}{16})$, con $0 \leq i, j \leq 7$.

Debido a estas observaciones, se tiene que la transformada DCT de una matriz B concentra la información de ésta en las coordenadas cercanas al ángulo superior izquierdo, mientras que las otras coordenadas tomaran valores cercanos a cero.

Ejemplo 1. Supongamos que la matriz B tiene como entradas sólo los números $50, 51, 52$,

$$B = \begin{pmatrix} 51 & 52 & 51 & 50 & 50 & 52 & 50 & 52 \\ 51 & 52 & 51 & 51 & 50 & 52 & 52 & 51 \\ 50 & 50 & 51 & 52 & 52 & 51 & 51 & 51 \\ 51 & 50 & 50 & 50 & 52 & 50 & 50 & 51 \\ 51 & 50 & 50 & 51 & 50 & 50 & 51 & 50 \\ 50 & 51 & 52 & 52 & 51 & 50 & 50 & 50 \\ 51 & 52 & 51 & 50 & 52 & 50 & 52 & 50 \\ 50 & 51 & 52 & 52 & 50 & 51 & 52 & 51 \end{pmatrix}$$

En este caso la matriz resultante al aplicarle la transformada DTC queda

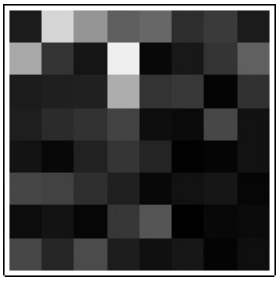
$$U = \begin{pmatrix} 407 & 0{,}058 & -0{,}518 & -0{,}592 & -0{,}5 & 0{,}118 & -0{,}597 & 0{,}086 \\ 0{,}352 & -0{,}654 & 1{,}019 & 0{,}818 & 0{,}179 & -1{,}074 & 1{,}190 & -1{,}194 \\ 1{,}904 & -0{,}116 & 1 & -0{,}598 & -2{,}174 & -0{,}352 & 0{,}293 & -1{,}006 \\ -0{,}661 & 1{,}350 & 0{,}689 & -0{,}055 & -0{,}425 & -0{,}599 & 0{,}254 & -0{,}412 \\ -1 & -0{,}335 & 1{,}171 & 0{,}102 & 0{,}5 & -0{,}020 & 0{,}868 & -0{,}502 \\ -0{,}229 & 0{,}162 & 0{,}115 & 0{,}711 & 0{,}956 & -1{,}902 & -0{,}108 & 1{,}454 \\ 0{,}023 & -0{,}173 & -1{,}707 & -1{,}529 & 0{,}630 & 0{,}109 & 1 & -0{,}603 \\ -0{,}110 & -0{,}383 & 0{,}105 & 0{,}470 & 0{,}005 & 0{,}568 & -0{,}470 & 0{,}111 \end{pmatrix}.$$

Ahora, si consideramos el bloque de la Figura 3, en este caso la matriz resultante es

$$U = \begin{pmatrix} -27{,}50 & -213{,}47 & -149{,}61 & -95{,}28 & -103{,}75 & -46{,}95 & -58{,}72 & 27{,}23 \\ 168{,}23 & 51{,}61 & -21{,}54 & -239{,}52 & -8{,}24 & -24{,}49 & -52{,}66 & -96{,}62 \\ -27{,}20 & -31{,}24 & -32{,}28 & 173{,}39 & -51{,}14 & -56{,}94 & 4{,}00 & 49{,}14 \\ 30{,}18 & -43{,}07 & -50{,}47 & 67{,}13 & -14{,}12 & 11{,}14 & 71{,}01 & 18{,}04 \\ 19{,}50 & 8{,}46 & 33{,}59 & -53{,}11 & -36{,}75 & 2{,}92 & -5{,}79 & -18{,}39 \\ -70{,}59 & 66{,}88 & 47{,}44 & -32{,}61 & -8{,}19 & 18{,}13 & -22{,}99 & 6{,}63 \\ 12{,}08 & -19{,}13 & 6{,}25 & -55{,}16 & 85{,}59 & -0{,}60 & 8{,}03 & 11{,}21 \\ 71{,}15 & -38{,}37 & -75{,}92 & 29{,}29 & -16{,}45 & -23{,}44 & -4{,}21 & 15{,}62 \end{pmatrix}.$$

En ambos casos hemos redondeado los resultados 3 y 2 dígitos, respectivamente.

A continuación mostramos el bloque y la imagen original transformada, respectivamente.



Bloque transformado.

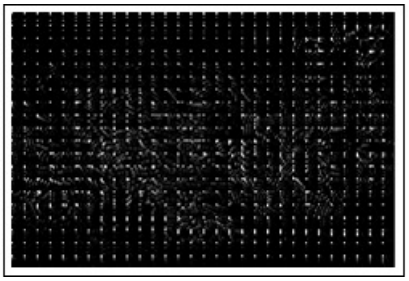


Imagen original transformada.

Como podemos observar las entradas de la matriz B, del bloque de la Figura 3, luego de la etapa de preprocesamiento, son números enteros, mientras que la transformada DCT convierte dichas entradas generalmente en números reales. El proceso de cuantificación se encarga de convertir los mismos en enteros como veremos a continuación.

§3. Cuantificación

El proceso de cuantificación consiste en dividir cada entrada de la matriz U por los elementos de la siguiente matriz de cuantificación Q.

$$Q = \begin{pmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{pmatrix}.$$

Para convertir los números resultantes en números enteros creamos la siguiente matriz $Q' = (Q'_{jk})$ definida por

$$Q'_{jk} = \left[\frac{U_{jk}}{Q_{jk}}\right] \qquad j, k = 1, ..., 8,$$

donde $[\cdot]$ denota la función que redondea al entero más próximo. Observemos que los valores de la esquina inferior derecha de Q son más grandes, al dividir por dichos números se obtienen valores cercanos al cero, por lo cual al redondearlos se convierten en 0.

Ejemplo 2. Si aplicamos el proceso de cuantificación al bloque de la Figura 3 tenemos

$$Q' = \begin{pmatrix} -2 & -19 & -15 & -6 & -4 & -1 & -1 & 0 \\ 14 & 4 & -2 & -13 & 0 & 0 & -1 & -2 \\ -2 & -2 & -2 & 7 & -1 & -1 & 0 & 1 \\ 2 & -3 & -2 & 2 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & -1 & -1 & 0 & 0 & 0 \\ -3 & 2 & 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 \\ 1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

que corresponde al bloque

Figura 5. Bloque cuantificado.

Mientras que aplicar la transformada del coseno discreta (DCT) y el proceso de cuantificación a la imagen original da las figuras

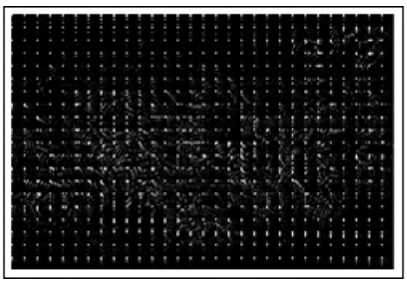


Imagen original transformada.

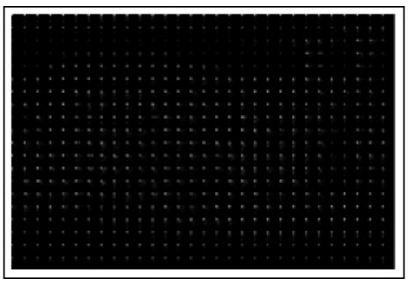


Imagen original cuantificada.

Recordemos que la transformada DCT es inversible. Como C es ortogonal, $C^t = C^{-1}$, por lo tanto

$$DCT(B)^{-1} = (CBC^t)^{-1} = (C^t)^{-1}B^{-1}C^{-1} = CB^{-1}C^t = DCT(B^{-1}).$$

Además, la división por un número es inversible, mientras que el redondeo no, por eso el formato JPEG es un ejemplo de compresión con pérdida.

La compresión con pérdida se refiere a cualquier procedimiento de codificación que tenga como objetivo representar cierta cantidad de información utilizando una menor cantidad de la misma, siendo imposible una reconstrucción exacta de los datos originales. Esto es porque, en lugar de guardar una copia exacta, solo se guarda una aproximación. Esta aproximación se aprovecha de las limitaciones de la percepción humana para esconder la distorsión introducida.

§4. Codificación (Huffman)

El último paso en el proceso JPEG es codificar la imagen transformada y cuantificada. El estándar de compresión JPEG usa una versión avanzada del código de Huffman ([8]), siguiendo un recorrido en zig-zag en la matriz para obtener una lista con los ceros acumulados al final (ver Figura 6).

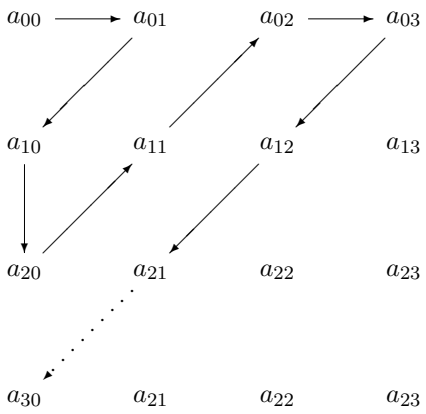


Figura 6. Lista $(a_{00}, a_{01}, a_{10}, a_{20}, a_{11}, \ldots)$ obtenida con codificacion de Hufmann.

En nuestro ejemplo

$$Q' = \begin{pmatrix} -2 & -19 & -15 & -6 & -4 & -1 & -1 & 0 \\ 14 & 4 & -2 & -13 & 0 & 0 & -1 & -2 \\ -2 & -2 & -2 & 7 & -1 & -1 & 0 & 1 \\ 2 & -3 & -2 & 2 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & -1 & -1 & 0 & 0 & 0 \\ -3 & 2 & 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 \\ 1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

La lista $(a_{00}, a_{01}, a_{10}, a_{20}, a_{11}, \ldots)$ para esta matriz es

$$\begin{array}{lllllll} a_{00} = -2, & a_{01} = -19, & a_{10} = 14, & a_{20} = -2, & a_{11} = 4, & a_{02} = -15, & a_{03} = -6, \\ a_{12} = -2, & a_{21} = -2, & a_{30} = 2, & a_{40} = 1, & a_{31} = -3, & a_{22} = -2, & a_{13} = -13, \\ a_{04} = -4, & a_{05} = -1, & a_{14} = 0, & a_{23} = 7, & a_{32} = -2, & a_{41} = 0, & a_{50} = -3, \\ a_{60} = 0, & a_{51} = 2, & a_{42} = 1, & a_{33} = 2, & a_{24} = -1, & a_{15} = 0, & a_{06} = -1, \\ a_{07} = 0, & a_{16} = -1, & a_{25} = -1, & a_{34} = 0, & a_{43} = -1, & a_{52} = 1, & a_{61} = 0, \\ a_{70} = 1, & a_{71} = 0, & a_{62} = 0, & a_{53} = -1, & a_{44} = -1, & a_{35} = 0, & a_{26} = 0, \\ a_{17} = -2, & a_{27} = 1, & a_{36} = 1, & a_{45} = 0, & a_{54} = 0, & a_{63} = -1, & a_{72} = -1, \end{array}$$

y finalmente $a_{73} = 0$, $a_{64} = 1$, F, donde la letra F indica que a partir de ese elemento a_{55} hasta el final de la lista, de 64 elementos, son todos ceros. Posteriormente se codifica cada lista con una versión avanzada del código de Huffman (ver [5]).

La imagen original tiene dimensión 160×240 pixeles de manera que se necesitan $160 \times 240 \times 8 = 307{,}200$ bits para almacenarla en el disco. Si aplicamos la codificación de Huffman (ver [5]) para la versión transformada y cuantificada de la imagen de nuestro ejemplo en particular, necesitamos sólo $85{,}143$ bits para almacenar la imagen en el disco. La tasa de compresión es de aproximadamente $2{,}217$bpp (Bits por pixel). Esto representa un ahorro de más del $70\,\%$ de la cantidad original de bits necesarios para almacenar la imagen.

Para invertir el proceso y obtener una aproximación del bloque de la imagen original, el primer paso, trabajando por bloques 8×8, es decodificar el código de Huffman para obtener la matriz cuantificada del bloque trabajado. Posteriormente se invierte el paso de la división y luego aplicamos la inversa de la transformada DCT, C, y obtenemos una aproximación del bloque original preprocesado U, U'. Es decir, formamos primero la matriz

$$U'_{jk} = Q'_{jk} * Q_{jk}.$$

Ejemplo 3. Si invertimos el proceso de división obtenemos

$$U' = \begin{pmatrix} -32 & -209 & -150 & -96 & -96 & -40 & -51 & 0 \\ 168 & 48 & -28 & -247 & 0 & 0 & -60 & -110 \\ -28 & -26 & -32 & 168 & -40 & -57 & 0 & 56 \\ 28 & -51 & -44 & 58 & 0 & 0 & 80 & 0 \\ 18 & 0 & 37 & -56 & -68 & 0 & 0 & 0 \\ -72 & 70 & 55 & -64 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -87 & 103 & 0 & 0 & 0 \\ 72 & 0 & -95 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

mientras que nuestro bloque U original era

$$U = \begin{pmatrix} -27{,}50 & -213{,}47 & -149{,}61 & -95{,}28 & -103{,}75 & -46{,}95 & -58{,}72 & 27{,}23 \\ 168{,}23 & 51{,}61 & -21{,}54 & -239{,}52 & -8{,}24 & -24{,}49 & -52{,}66 & -96{,}62 \\ -27{,}20 & -31{,}24 & -32{,}28 & 173{,}39 & -51{,}14 & -56{,}94 & 4{,}00 & 49{,}14 \\ 30{,}18 & -43{,}07 & -50{,}47 & 67{,}13 & -14{,}11 & 11{,}14 & 71{,}01 & 18{,}04 \\ 19{,}50 & 8{,}46 & 33{,}59 & -53{,}11 & -36{,}75 & 2{,}92 & -5{,}79 & -18{,}39 \\ -70{,}59 & 66{,}88 & 47{,}44 & -32{,}61 & -8{,}19 & 18{,}13 & -22{,}99 & 6{,}63 \\ 12{,}08 & -19{,}13 & 6{,}25 & -55{,}16 & 85{,}59 & -0{,}60 & 8{,}03 & 11{,}21 \\ 71{,}15 & -38{,}37 & -75{,}92 & 29{,}29 & -16{,}45 & -23{,}44 & -4{,}21 & 15{,}62 \end{pmatrix}$$

Observemos que los valores de U' difieren un poco de los valores correspondientes a U.

Luego, obtenemos una aproximación B' de B aplicando la transformada DCT en sentido inverso, $B' = C^t U' C$

$$B' = \begin{pmatrix} -128 & 45 & 71 & 63 & 22 & 32 & 22 & 52 \\ -110 & 39 & 4 & 48 & 41 & 68 & 65 & 13 \\ -115 & 40 & 50 & 152 & 13 & -88 & -4 & 76 \\ -105 & 51 & 43 & 18 & -126 & -99 & 19 & 60 \\ -116 & -24 & 56 & 63 & 33 & 80 & 62 & 28 \\ -67 & -118 & -47 & -24 & 30 & 17 & -12 & 29 \\ -67 & -79 & -60 & -116 & 49 & 69 & 12 & -108 \\ -78 & -69 & -80 & -138 & 63 & 41 & 49 & -67 \end{pmatrix}$$

Observar la diferencia con la matriz original B, y su consiguiente pérdida de información.

$$D = B' - B = \begin{pmatrix} 6 & -4 & 5 & 22 & -19 & -11 & -18 & 14 \\ 11 & 10 & -27 & 3 & 6 & 18 & 24 & -11 \\ 7 & 0 & 5 & 47 & -18 & -22 & -22 & -11 \\ -11 & -1 & -1 & -29 & -4 & 23 & 11 & 9 \\ 3 & -1 & 3 & 12 & -12 & 10 & 1 & -14 \\ -3 & 4 & -22 & 2 & -3 & 2 & -18 & 17 \\ 9 & 1 & 4 & 6 & -4 & 5 & -26 & 14 \\ 0 & 5 & 4 & -16 & 6 & -3 & 8 & -14 \end{pmatrix}$$

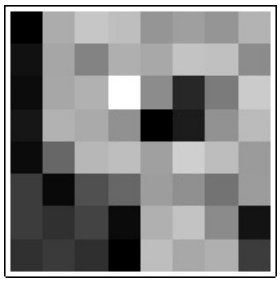
El último paso es sumar 127 a cada elemento del bloque. La matriz resultante es una aproximación del bloque original de la imagen original. El bloque resultante es

$$R = \begin{pmatrix} 0 & 172 & 198 & 190 & 149 & 159 & 149 & 179 \\ 17 & 166 & 131 & 175 & 168 & 195 & 192 & 140 \\ 12 & 167 & 177 & 255 & 140 & 39 & 123 & 203 \\ 22 & 178 & 170 & 145 & 1 & 28 & 146 & 187 \\ 11 & 103 & 183 & 190 & 160 & 207 & 189 & 155 \\ 60 & 9 & 80 & 103 & 157 & 144 & 115 & 156 \\ 60 & 48 & 67 & 11 & 176 & 196 & 139 & 19 \\ 49 & 58 & 47 & 0 & 190 & 168 & 176 & 60 \end{pmatrix}$$

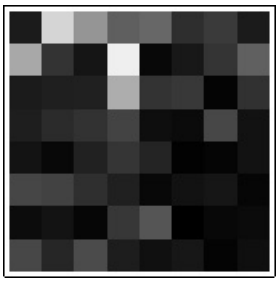
y el bloque original es

$$F = \begin{pmatrix} 5 & 176 & 193 & 168 & 168 & 170 & 167 & 165 \\ 6 & 176 & 158 & 172 & 162 & 177 & 168 & 151 \\ 5 & 167 & 172 & 232 & 158 & 61 & 145 & 214 \\ 33 & 179 & 169 & 174 & 5 & 5 & 135 & 178 \\ 8 & 104 & 180 & 178 & 172 & 197 & 188 & 169 \\ 63 & 5 & 102 & 101 & 160 & 142 & 133 & 139 \\ 51 & 47 & 63 & 5 & 180 & 191 & 165 & 5 \\ 49 & 53 & 43 & 5 & 184 & 170 & 168 & 74 \end{pmatrix}$$

mientras que las imágenes correspondientes son



Bloque resultante.



Bloque original.

Las imágenes a continuación muestran la imagen original a la izquierda y la imagen JPEG comprimida a la derecha. Recordemos que la imagen de la derecha requiere menos del 70 % de espacio de almacenamiento que el de la imagen original.

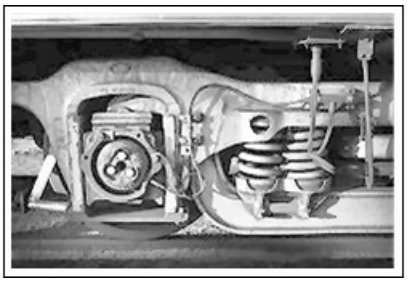


Imagen original.

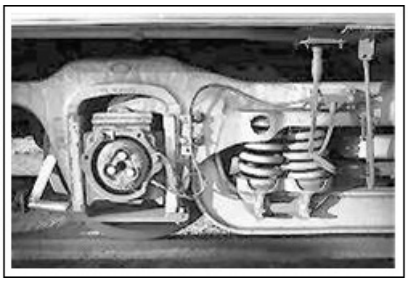
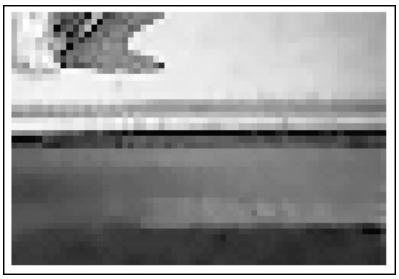


Imagen resultante.

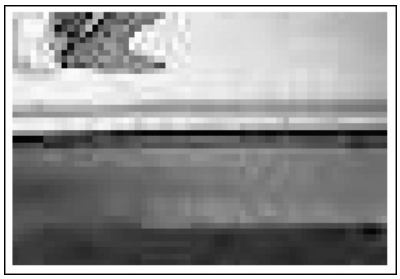
§5. Cuestiones y problemas

En esta sección trataremos y mencionaremos algunos de los problemas o inconvenientes que presenta el estándar JPEG. Primero, como mencionamos anteriormente, la matriz C de la transformada DCT satisface, $C^t = C^{-1}$. La división por un número es reversible. Sin embargo, la conversión de los pequeños valores a 0 y el redondeo de todos los valores cuantificados no son pasos reversibles. Perdemos la capacidad de recuperar la imagen original, por eso el JPEG es un ejemplo de compresión con pérdida. Otro aspecto a observar es que al dividir la imagen en bloques de 8×8 se produce una disociación entre bloque y bloque perdiendo la continuidad de la imagen, que se suele mostrar en la imagen resultante los bloque 8×8 bien remarcados y generando artefactos en la misma. Este problema

lo podemos ver en el siguiente ejemplo de la esquina superior derecha de la imagen original y de la imagen comprimida. Para poder apreciar los mismos las imágenes se muestran con zoom.



Esquina superior derecha
Imagen original.



Esquina superior derecha
Imagen comprimida.

Teniendo en cuentas los inconvenientes anteriormente mencionados, en 1997, el Joint Photographic Experts Group (JPEG) comenzó a trabajar en la próxima versión del estándar de compresión de imágenes JPEG. La nueva norma estándar fue nombrada JPEG 2000. Al igual que JPEG, JPEG2000 consta de cuatro pasos básicos en el algoritmo, preprocesamiento, transformación, cuantificación y codificación. A diferencia de JPEG, la etapa de cuantificación es opcional si el usuario desea realizar la compresión sin pérdidas, primera gran diferencia con JPEG. Mientras que JPEG utiliza una versión avanzada de la codificación de Huffman, JPEG2000 utiliza un nuevo método de codificación llamado Embedded codificación de bloque con truncamiento optimizado (EBCOT). Uno de los principales cambios en el estándar JPEG2000 es el uso de la transformada wavelet discreta (DWT) en lugar de la DCT.

El estándar JPEG sigue siendo el estándar más popular para el almacenamiento de imágenes en servidores web y la resolución con la que las imágenes se comprimen utilizando la norma es aceptable para aplicaciones tales como la navegación web. A medida que continúa la era digital, las personas que trabajan con imágenes digitales identificaron algunos problemas con JPEG, así como algunos servicios que podrían mejorar aún más la norma. A continuación mencionamos algunos de ellos:

- *Sin opción de compresión sin pérdidas*. Para muchas aplicaciones (anuncios de revistas, por ejemplo), la pérdida de resolución no es aceptable.
- *El desacoplamiento de Imágenes*. Partición de la imagen en 8×8 bloques significa que los bloques son vistos de manera independiente - que se transforman y cuantifican independientes entre sí. Esto es una desventaja cuando se trata de explotar la homogeneidad de una región que puede ser mayor que un bloque de 8×8.
- *Efectos de bloques*. Dado que la imagen se divide en bloques, con frecuencia no hay una transición suave entre los bloques en la imagen comprimida.
- *Los efectos de borde*. La DCT se construye a partir de muestras de la función coseno y por lo tanto funciona mejor cuando los datos de entrada son periódicos. No es el caso típico de que las intensidades en filas o columnas de una imagen digital son periódicas.

La mayor parte de los problemas encontrados por la DCT en JPEG se resolvieron con la incorporación de la Transformación Discreta Wavelet en JPEG2000. Características y mejoras del estándar de compresión de imagen de JPEG2000:

- *Mejor Compresión*. En muchos casos, fijamos la tasa de compresión y luego ajustamos el algoritmo en consecuencia. El JPEG2000 produce imágenes de mayor calidad para bajos radios de compresión de bits (0,025bpp y menor) que JPEG.
- *Transmisión de señal progresiva*. JPEG2000 puede reconstruir imágenes digitales (a un ritmo creciente de la resolución) a medida que se reciben por un navegador.
- *Bloque*. JPEG2000 permite a los usuarios elegir dividir la imagen en bloques mas grandes que los de tamaño 8×8.
- *Regiones de interés*. Los usuarios de JPEG2000 pueden identificar regiones de interés, ROI, en una imagen y codificar estás ROIs para que tengan una mayor resolución en la versión comprimida de la imagen.
- *Tamaño de imagen*. El estándar JPEG sólo puede manejar imágenes de tamaño hasta 64.000×64.000, mientras que JPEG2000 puede lidiar con imágenes de tamaño $4.294.967.295 \times 4.294.967.295$ ($4.294.967.295 = 2^{32} - 1$).

Referencias

[1] Charrier M., Cruz D. S., and Larsson M. *JPEG2000, the next millennium compression standard for still images*. Multimedia Computing and Systems, 1999. IEEE International Conference on (Volume: 1 pages 131 - 132) Jul 1999.

[2] Gailly J., Nelson M. *The Data Compression Book*. Second edition. M&T Books. 1996.

[3] Lezama J. *Image compression by Johnson graphs*. Proceeding of 2015 XVI Workshop on Information Processing and Control (RPIC). IEEE-Explore. ISBN 978-1-4673-8466-7, 2015.

[4] Lezama J. *Compresión de imágenes y cálculo eficiente de proyecciones a autoespacios*. Tesis de Doctordado, Publicaciones C de FaMAF (UNC), Mayo de 2013.

[5] Lezama J. *Compresión de imágenes - Codificación de Huffman*. Revista de Educación Matemática, Vol. 32, N° 1, (25–37), 2017.

[6] Levstein F., Lezama J., Maldonado C., Penazzi, D., Schilman M. *Hamming Graph based Image Compression with variable Threshold*, GVIP Journal, ISSN: 1687-398X, Volume 15, Issue 1, ICGST, Delaware, USA, 2015.

[7] Strang G. *The Discrete Cosine Transform* , Journal SIAM Review Volume 41 Issue 1, (135–147), 1999.

[8] Image Compression. http://www.whydomath.org/node/wavlets/index.html. SIAM.

Javier Lezama
Facultad de Matemática, Astronomía, Física y Computación (Famaf), *Universidad Nacional de Córdoba* (UNC).
Av. Medina Allende s/n , Ciudad Universitaria (X5000HUA) Córdoba, Argentina.
(✉) javitolez@gmail.com

Recibido: *9 de enero de 2016*.
Aceptado: *15 de abril de 2016*.
Publicado en línea: *11 de octubre de 2017*.

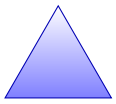
¿Sabías que...?

por Ricardo Podestá

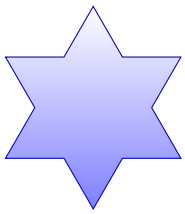
¡Existen figuras planas de área finita y acotada con perímetro infinito!

¡Increíble! A primera vista esto parece imposible, pero veremos que no lo es. En efecto, construiremos una sucesión infinita $\{K_n\}_{n=0}^{\infty}$ de polígonos de área finita y acotada tales que sus perímetros tienden a infinito. La figura plana que se obtenga como límite de esta sucesión de polígonos será por tanto la figura deseada. Sorprendentemente, para ver esto sólo necesitamos usar el área de un triángulo y la serie geométrica.

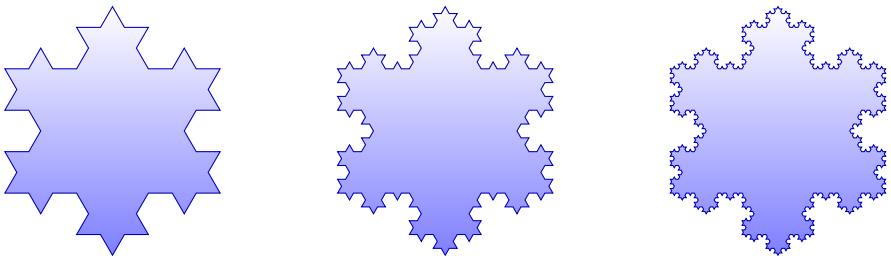
La sucesión de polígonos. Tomemos como K_0 a un triángulo equilátero de lado 1



y, a partir de éste, construimos el polígono K_1 de la siguiente manera. A cada lado de K_0 lo dividimos en 3 segmentos iguales, y sobre cada uno de los 3 segmentos medios (que no tocan vértices de K_0) apoyamos un triángulo equilátero de lado $\frac{1}{3}$ y removemos los segmentos comunes en donde se intersecan el triángulo grande con los pequeños. El polígono K_1 luce así



Para obtener K_2 procedemos análogamente adjuntando, a cada segmento medio de longitud $\frac{1}{9}$, triangulitos equiláteros de lado $\frac{1}{9}$ y removiendo los lados que se intersecan. Iterando este procedimiento, para cada n natural podemos construir recursivamente un nuevo polígono K_{n+1} a partir del anterior K_n. Los polígonos K_2, K_3 y K_4 tienen la pinta



Cada uno de los polígonos no convexos K_n se llama *copo de nieve de Koch* y presentan una complejidad creciente. Por ejemplo, estos copos de Koch son un paradigma de las figuras fractales. Si iteramos infinitamente el procedimiento descripto, se obtiene como límite una figura plana que denotaremos por K y que no es un polígono (¿porqué?). El borde de K es conocido como la *curva de Koch*. Esta curva plana es continua, pero sin embargo ¡no tiene tangentes bien definidas en ningún punto!

Perímetro infinito de K. Notar que los polígonos K_n están formados por segmentos de igual longitud. Luego, el perímetro de K_n está dado por el número de lados de K_n, digamos s_n, multiplicado por la longitud de cada lado, digamos ℓ_n, es decir

$$\text{Per}(K_n) = s_n \cdot \ell_n.$$

La longitud de K_n es, por construcción, $\ell_n = \frac{1}{3^n}$. En cada paso del proceso de construcción, cambiamos un lado de K_n por 4 lados mas pequeños en K_{n+1}, por lo que el número de lados es 3, $12 = 3 \cdot 4$, $48 = 3 \cdot 4^2$, etc. Luego, el número de lados de K_n es $s_n = 3 \cdot 4^n$. De esta manera, el perímetro de K_n es

$$\text{Per}(K_n) = 3 \cdot 4^n \cdot \tfrac{1}{3^n} = 3(\tfrac{4}{3})^n.$$

Luego, la longitud L de la curva de Koch es el límite de los perímetros de los K_n cuando n se hace infinitamente grande. En símbolos,

$$L = \lim_{n\to\infty} \text{Per}(K_n) = \lim_{n\to\infty} 3e^{n\log(\frac{4}{3})} = \infty$$

como queríamos ver.

A modo de ejemplo, supongamos que nuestro triángulo inicial K_0 es de 1 decímetro (10 cm = 0.0001 km) de lado, es decir cabe en nuestras manos. Tenemos, $\text{Per}(K_5) = 12,64$, o sea que ya supera el metro. Mas sorprendentemente, tenemos

n	Per(K_n) en [dm]	mayor que
21	1261,35	cancha de fútbol
29	12599,25	altura del Uritorco
57	39685261,05	longitud de Argentina
66	528540012,78	circunferencia de la Tierra

lo cual nos da una idea de lo ¡rápido que crece el perímetro!

Área finita y acotada de K. El polígono K_n es construido adjuntando triángulos equiláteros mas pequeños de distintos tamaños al triángulo K_0. Por lo tanto el área de K_n es la suma del área de todos estos triangulitos. Debemos saber cuántos triángulos de cada tamaño hay y cuál es el área de cada uno de ellos. Luego, el área de K_n es

$$A(K_n) = \sum_{k=0}^{n} N_k \cdot A(T_k)$$

donde N_k es el número de triángulos equiláteros T_k de lado $b_k = \frac{1}{3^k}$ adjuntados en el paso k de la construcción de la curva K (aquío $T_0 = K_0$).

Ahora, el área de un triángulo T de base b y altura h es $\frac{bh}{2}$. Si éste es equilátero, la altura divide a un lado en 2 y usando el teorema de Pitagoras podemos calcular que la altura es $h = \frac{\sqrt{3}}{2}b$ y, por lo tanto, su área es $A(T) = \frac{\sqrt{3}}{4}b^2$. De esta manera, el área de T_k es

$$A(T_k) = \tfrac{\sqrt{3}}{4}(\tfrac{1}{3^k})^2.$$

En el paso 1 adjuntamos 3 triángulos T_1 a K_0, en el paso 2 adjuntamos $3 \cdot 4$ triángulos T_2 a K_1, en el paso 3 adjuntamos $3 \cdot 4^2$ triángulos T_3 a K_2 etc. Luego,

$$N_k = 3 \cdot 4^{k-1}$$

para $k \geq 1$ y por lo tanto tenemos

$$A(K_n) = \frac{\sqrt{3}}{4} + \frac{3\sqrt{3}}{4}\sum_{k=1}^{n} 4^{k-1}(\frac{1}{3^k})^2 = \frac{\sqrt{3}}{4} + \frac{3\sqrt{3}}{16}\sum_{k=1}^{n}(\frac{4^k}{3^k})^2 = \frac{\sqrt{3}}{4}\{1 + \frac{3}{4}\sum_{k=1}^{n}(\frac{4}{9})^k\}.$$

El área A de la curva de Koch será pues el límite de las áreas de los polígonos K_n a medida que n tiende a infinito. En símbolos,

$$A = \lim_{n\to\infty} A(K_n) = \frac{\sqrt{3}}{4}\{1 + \frac{3}{4}\sum_{k=1}^{\infty}(\frac{4}{9})^k\}.$$

La serie (suma infinita) que nos queda se puede calcular. De hecho, es la conocida serie geométrica; si $0 < s < 1$ vale $\sum_{k=1}^{\infty} r^k = \frac{r}{1-r}$. Usando esto con $r = \frac{4}{9}$ tenemos que

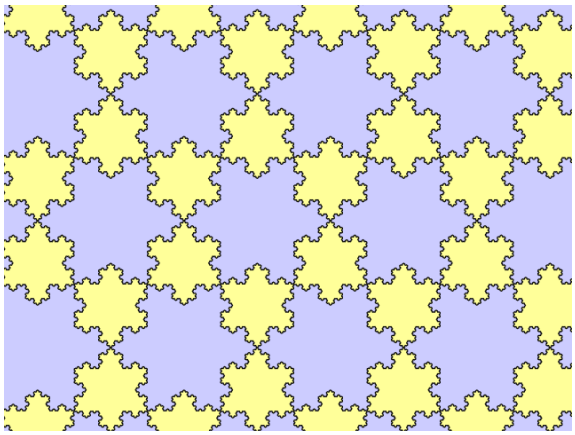
$$\sum_{k=1}^{\infty} r^k = \frac{\frac{4}{9}}{1-\frac{4}{9}} = \frac{4}{5}.$$

Finalmente,

$$A = \frac{\sqrt{3}}{4}(1 + \frac{3}{4}\cdot\frac{4}{5}) = \frac{8}{5}\cdot\frac{\sqrt{3}}{4}.$$

O sea, no solo vemos que el área de la figura contenida dentro de K es finita, sino que es ¡menos del doble del área del triángulo K_0 con el que partimos!

Una curiosidad. Por último, no queremos dejar de mencionar, que es posible teselar el plano (cubrir completamente sin intersecciones) con copos de nieve de Koch de dos tamaños diferentes. No tenemos lugar aquí para una demostración, pero dejamos una imagen para regocijo de nuestros ojos



(Imagen tomada del blog "Complex projective 4-space")

Finalmente, cerramos con una pregunta ¿Cuál es la circunsferencia de menor radio que contiene a la curva de Koch? (o sea, a todos los polígonos K_n).

Niels Fabian Helge von Koch fue un matemático sueco (Estocolmo, 25/1/1870 – 11/3/1924) que trabajaba en teoría de números. Descubrió la curva que hoy lleva su nombre.